Clmpto

Pln

1. A method of increasing the obscurity and tamper-resistance of a software program, comprising the steps of:

randomly generating substantive yet redundant arguments; and

inserting said arguments into the data flow of said program.

2. A method as claimed in claim 1, wherein said steps of randomly generating and inserting comprise the steps of:

randomly generating substantive yet redundant, lookup tables; and

inserting said lookup tables into the data flow of said program.

3. A method as claimed in claim 2, wherein said steps of randomly generating and inserting comprise the steps of:

introducing longitudinal diffusion by:

randomly generating identity look up tables; and

inserting said identity look up tables into the data flow of said program.

4. A method as claimed in claim 3, wherein said program is a data encryption standard (DES) program and said step of randomly generating comprises the step of randomly generating DES-based identities as networks of T-boxes.

5. A method as claimed in claim 4, wherein said DES-based identities comprise complementary encryption and decryption lookup tables containing a cryptographic key unlike the secret cryptographic key of said DES program.

6. A method as claimed in claim 5, wherein said step of inserting comprises the step of:

placing said DES-based identities before and after one or more initial round pairs of said DES program, and before and after one or more final round pairs, thereby defending against attacks from the ends of said DES program.

7. A method as claimed in claim 2, wherein said steps of randomly generating and inserting comprise the steps of:

splitting the data flow of said program into separate streams; and

diffusing data laterally between said separate streams.

8. A method as claimed in claim 2, wherein said steps of randomly generating and inserting comprise the steps of:

introducing lateral diffusion by:

generating multiple lookup tables for an original lookup table;

generating entries for said multiple lookup tables in accordance with a random Boolean function; and

transposing the output of said multiple lookup tables in accordance with said random Boolean function.

9. A method as claimed in claim 8, wherein said step of generating entries comprises:

choosing a random, substantive, Boolean function;

for each output of said original lookup table:

determining the set of inputs to said Boolean function that will yield said output of said original lookup table;

randomly selecting one of said sets of inputs; and

inserting said selected set of inputs, into the output of said multiple lookup tables;

modifying calls to said original lookup table to call upon said multiple lookup tables; and

inserting said random Boolean function into the data flow of said program following said calls to said multiple lookup tables.

10. A method as claimed in claim 8, wherein said random Boolean function is a two input Boolean function and each said set of inputs comprises two inputs.

11. A method as claimed in claim 9, wherein said random Boolean function is a three input Boolean function and each said set of inputs comprises three inputs.

12. (Amended) A method as claimed in claim 10, wherein said steps are executed beginning at penultimate lookup tables, and working backwards towards earlier rounds.

13. (Amended) A method as claimed in claim 6, wherein said software program is an encryption program requiring a cryptographic key, said method comprising the previous step of:

converting said software program into a direct acyclic graph.

14. A method as claimed in claim 13, wherein said encryption program is a Data Encryption Standard (DES) program and said step of converting comprises the steps of:

unrolling the *n* digital encryption software algorithm rounds by:

duplicating the round network *n* times and connecting said *n* rounds end-to-end;

copying the *i* S-boxes explicitly into each round, resulting in *n x i* separate S-boxes; and

converting each said *k*-output S-box into *k* 1-output T-boxes resulting in *n x i x k* separate T-boxes, with *k x i* separate T-boxes per round.

15. A method as claimed in claim 14, wherein said step of converting comprises the step of:

unrolling the sixteen digital encryption software algorithm rounds by:

duplicating the round network sixteen times and connecting said rounds end-to-end;

copying the eight S-boxes explicitly into each round, resulting in 128 separate S-boxes; and

converting each said 4-output S-box into four 1-output T-boxes resulting in 512 separate T-boxes, thirty-two per round.

16. A method as claimed in claim 15, further comprising the step of:

partially evaluating said program to eliminate the cryptographic key as a separate constant or series of constants.

17. A method as claimed in claim 16, further comprising the step of:

where one operand of an XOR operation adjacent to a T-box is a constant, eliminating said XOR operation by:
modifying the entries of said T-box to effect said XOR accordingly; and
deleting said XOR operation.

18. An apparatus for increasing the obscurity and tamper-resistance of computer software code comprising:
means for modifying said software code by:
randomly generating substantive yet redundant arguments; and
inserting said arguments into the data flow of said software code.

19. (Amended) A computer readable memory medium, storing computer software code executable to perform the steps of claim 1.

20. (Amended) A computer data signal embodied in a carrier wave, said computer data signal comprising a set of machine executable code being executable by a computer to perform the steps of claim 1.

21. The method as claimed in claim 1, wherein said substantive yet redundant arguments are substantive in that said arguments are applied to variables used in said software program, and that said arguments alter the value of said variables.

22. A method as claimed in claim 11, wherein said steps are executed beginning at penultimate lookup tables, and working backwards towards earlier rounds.

23. A method as claimed in claim 12, wherein said software program is an encryption program requiring a cryptographic key, said method comprising the previous step of:

converting said software program into a direct acyclic graph.

24. A method as claimed in claim 23, wherein said encryption program is a Data Encryption Standard (DES) program and said step of converting comprises the steps of:

unrolling the *n* digital encryption software algorithm rounds by:

duplicating the round network *n* times and connecting said *n* rounds

end-to-end;

copying the *i* S-boxes explicitly into each round, resulting in *n* x *i* separate

S-boxes; and

converting each said *k*-output S-box into *k* 1-output T-boxes resulting in

n x *i* x *k* separate T-boxes, with *k* x *i* separate T-boxes per round.

25. A method as claimed in claim 24, wherein said step of converting comprises the step of:

unrolling the sixteen digital encryption software algorithm rounds by:

duplicating the round network sixteen times and connecting said rounds

end-to-end;

copying the eight S-boxes explicitly into each round, resulting in 128 separate S-boxes; and

converting each said 4-output S-box into four 1-output T-boxes resulting in 512 separate T-boxes, thirty-two per round.

26. A method as claimed in claim 25, further comprising the step of:

partially evaluating said program to eliminate the cryptographic key as a separate constant or series of constants.

27. A method as claimed in claim 26, further comprising the step of:

where one operand of an XOR operation adjacent to a T-box is a constant,

eliminating said XOR operation by:

modifying the entries of said T-box to effect said XOR accordingly; and

deleting said XOR operation.